MPGM: A Mixed Parallel Big Graph Mining Tool

Ma Pengjiang[1]
mpjr_2008@163.com
Liu Yang[1]
liuyang1984@bupt.edu.cn

Wu Bin[1]
wubin@bupt.edu.cn
Wang Hongxu[1]
513196584@qq.com

[1]School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract—*The design and implementation of a scalable parallel mining system for big graph analysis has proven to be challenging. In this study, we propose a parallel data mining system for analyzing big graph data generated on a bulk synchronous parallel (BSP) computing model and MapReduce computing model named mixed parallel graph mining (MPGM). This system has four sets of parallel graph mining algorithms programmed in the BSP parallel model and one set of data extraction-transformation-loading (ETL) algorithms implemented in MapReduce and a well-designed workflow engine optimized for Cloud computing to invoke these algorithms. Experimental show that the components of graph mining algorithm in MPGM are efficient and can make realistic application easy.*

Keywords- Cloud computing; parallel algorithms; graph data analysis; data mining; social network analysis

1 Introduction

Graphs are the most widely used abstract data structures in the field of computer science, and they offer a more complex and comprehensive presentation of data compared to link tables and tree structures. Many real application issues need to be described with graphical structure, and the processing of graph data is required in almost all cases, such as the optimization of railway paths, prediction of disease outbreaks, the analysis of technical literature citation networks, emerging applications such as social network analysis, semantic network analysis, and the analysis of biological information networks.

The graph mining theories and technique have been improved all the time. However, as the information time comes along, which has led to explosive growth of information, the scale of graph-based data has increased significantly. For example, in recent decades, with the popularity of the Internet and the promotion of Web 2.0, the number of webpages has undergone rapid growth. Based on statistics provided by the China Internet Network Information Center (CNNIC), at the end of December 2013, the number of webpages in China had reached 150.0 billion, 22.2% increase over last year. Simultaneously, the number of micro-blog users accounted for 54.7% of all Internet users, which is approximately 308 million. This phenomenon highlights the scale of big graph data come into being, and it is challenging job to perform efficient analysis of these data. To solve the large scale graph analysis task, we have built a system called MPGM which provides a series of parallel graph mining algorithms based on the BSP parallel computing model. While the process of computing, the data is always stored in memory of cluster, this mechanism helps BSP model achieved a high performance, but limited the scale of data that the system can handle. Therefore, MPGM adds a set of data extract-transformation-loading algorithms based on MapReduce to improve the data processing capacity when the cluster scale is limited. Tests on real mobile communication networks data show that our improvement is reliable and highly- efficient.

The remainder of this paper is structured as follows. Section 2 reviews related works. And then describes MPGM's system architecture in Section 3. The application example is presented in Section 4. Some performance measurements are reported in Section 5. Finally we will discuss future directions.

2 Related Work

MGPM is closely related to parallel computing platforms and graph mining tools. Here, we briefly summarize those related works.

Parallel computing platforms have been studied for a long time, and they can be roughly categorized into three types: (i) based on the MapReduce model, (ii) based on the message passing interface (MPI) model, and (iii) based on the BSP model. The MapReduce [1] model was proposed by Google, and the most famous and successful open-source implementation is Hadoop. The MapReduce model is extreme suitable for process large scale data, and algorithms that do not have many iterations, but it has a bad performance in high iterative algorithm, which means that they are not suitable for most graph algorithms.

The MPI [2] model provides a model for message passing, and many companies and universities have implemented jobs that can be run on almost any type of parallel computer, which support all existing graph algorithms. However, because the MPI model uses a communication method to integrate computing resources, this model has several drawbacks, for example, the low efficiency of parallel computing and the high consumption of memory makes it difficult to manage the resources and communication in detail.

BSP [3] is also a widely used parallel computing framework. BSP improved the weakness exhibited by MapReduce, and performs well when a program has a large number of iterations or requires a lot of communication. A BSP program can be divided into several super-steps, each of which consists of three ordered stages: local computation, communication, and barrier synchronization. A BSP system is composed of a number of computers with local memory and disks. Each computer can run several computing processes called peers. In the local computation stage, each peer is computed using locally stored data. After finishing local computation, each peer can communicate only necessary data to other peers. When a peer finishes the communication stage, it will wait until all the peers reach the barrier synchronization and a super-step is completed.

Popular parallel data mining tools include the following things. Mahout [4], which is supported by the Apache Foundation, supply classification, clustering, pattern mining, regression, and dimension reduction and other machine learning algorithms, but lacks the graph mining function. GraphLab [5] improves on the MapReduce abstraction by compactly expressing asynchronous iterative algorithms with sparse computational dependencies. However, there may be problems while implied a synchronous iterative graph algorithm. PEGASUS [6] is an open-source large graph mining system implemented on Hadoop. The key idea of PEGASUS is to convert graph mining operations into iterative matrix-vector multiplication. While it supports large-scale graph data, in practice, not all of the graph mining algorithms can be modeled by matrix-vector multiplications. Dryad [7] is a general parallel computing platform proposed by Microsoft Research, which abstracts the computing and communication in data mining operations into vertexes and edges to form a dataflow graph. The platform executes the vertexes on work nodes and refines the dataflow graph to optimize the running process. Big Cloud parallel data mining (BC-PDM) [8] was developed by China Mobile Research Institute (CMRI), and it provides visualization operations for data mining and the analysis of graph data. However, it is based on Hadoop, and the graph mining algorithms therefore cannot achieve a high level of performance. Pregel [9], which was motivated by BSP and implemented by Google, provides a complete solution for large-scale graph computing, but it has not been published in the public domain. BC-BSP [10] is another implementation of the BSP parallel platform. While most BSP platforms use memory to exchange the temporary data, BC-BSP designed a mechanism of spill data (including static data and dynamical data) on the local disk to improve the data processing capacity when the cluster scale is limited, but the management and updating of this data spill mechanism requires extra communication and system resources, while introducing new defects to the platform.

3 System Architecture

This system focuses on big graph management and graph mining. We noticed that, while the original data is huge, but most graph mining application using part of the original data. In response to this feature, we use MapReduce to extract the graph data from original data, and construct the graph. To manage graph data and original data, we designed a data I/O management component in the parallel platform layer and a data management component in the logical layer. The algorithm layer divided into two parts, the ETL algorithm set and graph mining algorithm set. In the graph mining field, graph pattern mining, graph clustering mining, graph classification mining, and dynamic graph mining are the most popular topics. We built the graph mining algorithm set to implement graph clustering mining and graph classification mining algorithms, and the graph attribute analysis as the foundation of the graph analysis. Finally, we made this system extendable to enable the addition of other algorithm components.

An overview of the architecture of MPGM is presented in Figure 1. The system consists of four layers. The function of each layer is described as follows:

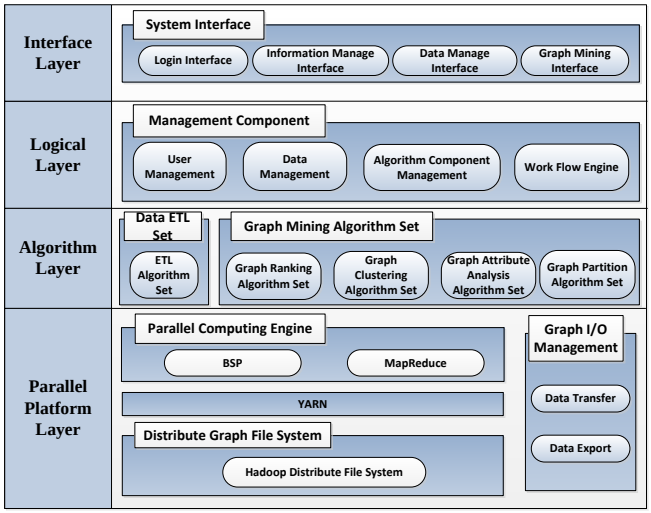

Figure 1: Architecture of PGM

3.1 The Parallel Platform Layer

The Parallel Platform Layer comprises four components: a Distributed Graph File System, YARN, Parallel Computing Engine, and Graph I/O Management component. We used the Hadoop Distributed File System (HDFS) to construct the Distributed Graph File System, enabling the storage of big graph data. YARN, a framework for cluster resource management and job scheduling, comprises a ApplicationMaster (AM) which should integrates multiple computing frameworks, e.g. MR, BSP. Because the BSP model achieves a high performance in graph mining algorithms, we chose Hama BSP [11] as the parallel computing engine, and used it to handle message communication, data distribution, and fault tolerance, and use MapReduce as the graph data pre-processing engine for extract graph information from original data. The Graph I/O Management component is responsible for the transfer of data from the database into the graph data form that the MPGM can handle, and it then exports the resulting MPGM data.

3.2 The algorithm layer

The algorithm layer is the main layer of MPGM. This layer can be roughly divided into 2 parts. The ETL algorithm set and graph mining algorithm set. In the graph mining algorithm set, we implemented four sets of 20 graph mining algorithm components in the BSP parallel model and four group of data ETL algorithm for transform original into graph data. The ETL algorithm set is composed of data cleaning set for detect and remove error value, data transform set for transform value into the format we need, data extract set and data update set. Those graph mining algorithm components can be divided into four sets. The graph ranking set comprises PageRank [12], HITS [13], and RWR [14] algorithms components, the graph clustering set comprises GN [15], CNM [16], CPM [17], and LPA [18] algorithm components, and the K-means algorithm is used for the processing of general data. The graph attribute analysis set contains the graph diameter, closeness centrality, clustering coefficient, network density, betweenness centrality, and five other algorithm components, while the graph partition set contains components that are based on the MSP [19] algorithm component and the Metis [20] algorithm component. Those algorithm components can be run either in the console, or can be invoked in a user-defined workflow from the user interface.

3.3 The logical layer

The logical layer is based on Open Service Gateway Initiative(OSGi), to implement a stable and efficient scalable system, which is service platform and manager all kinds of services that are supported by this layer, such as User Management Service, Data(HDFS) Management Service , Algorithm Service (Each algorithm is a kind of service), and the workflow engine Service.

Figure 2 shows the operation of the logical layer. Getting the start command from tomcat servlet which is a container in the interface layer, OSGi container will execute a train of operations, starting the life cycle, activating and registering the each service that hosted in bundle-services management. With the tomcat servlet invoking one of services, Service Register queries and gets service that is requested by the tomcat from the services pool, and then Execution Environment (EE) calls the workflow engine service to perform the requested service which is ultimately implemented in the algorithm layer. There are two important features:

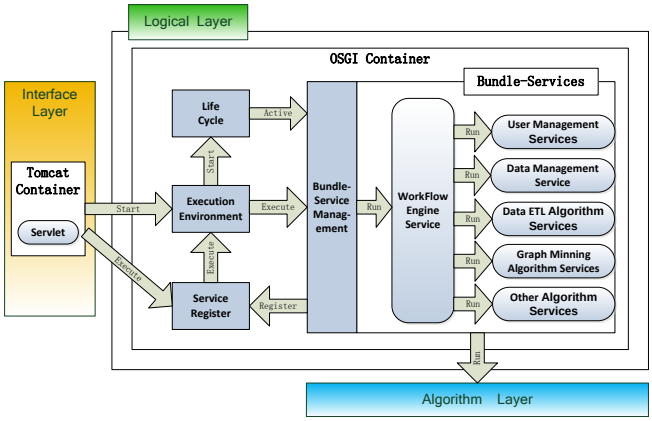

Figure 2: Operation flow of the Logical Layer

3.3.1 Hot- plugging of Bundle-Services.

Each service that provided by Bundle-Services Management is a plug-in or bundle with highly cohesion, low coupling features. With the independent class loader, OSGi prevents the external system accessing to detail of the bundle, just exposing externally callable interfaces, and provides a dynamic service management strategy. In other words, receiving the operation from the interface layer, Execution-Environment can dynamically install and uninstall the bundle-services, without shutting down the system.

3.3.2 High scalability and flexibility of Workflow Engine.

The workflow engine implemented in this system abstracts data-intensive computing into an orderly and plain workflow instance. Meanwhile, the workflow engine defining a set of unified interfaces can be seamless integrated with multiple computing frameworks, e.g. MR, BSP. Figure 3 shows an example workflow instance which includes three algorithms that can be divided into two categories, based on

MR and based on BSP. When the interface layer sends a command to execute, Workflow Engine automatically analyzes the workflow and orderly executes the each component. Workflow Engine also achieve computing operation monitoring and provides an flexible configuration .

3.4 The interface layer

The interface layer is built in HTML, and flex provides an interactive interface with which the user can login to the MPGM, management information, and most importantly, use the graph algorithms' mining graph data.

4 Application Example

Here we use the key user of mobile communication discovery as an example to demonstrate our plat-form.

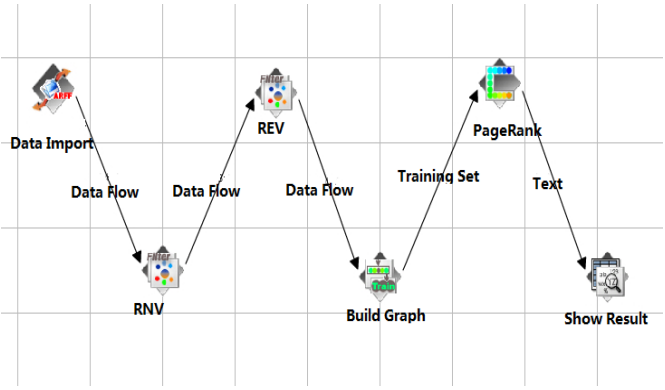

Figure 3: Operation flow of the Key User Discovery application

The discovery of key person in mobile communication is an important and valuable application. It is trying to find a number of users that influence people around them called the key user from communication information data with graph mining algorithm. The experience proven that delivery advertisement or making market strategy direct to those key users is more effective. The graph can be constructed from user mobile phone call record, and apply PageRank algorithm on the graph data, then the user with higher rank value has the higher influence to other users which means a key user.

However, in realistic application there is no purely graph data to make PageRank running on it, and the scale may be too large for our BSP platform on the cluster. The call information record has number of call and called user, the phone duration the phone happening time and data, some of user number cloud be null as they belong to other mobile operator. So we need to clean the data, select the call duration longer than 5 seconds, the less may be spam call, and get the purely graph data. The operation flow of this application in our platform is showed in Figure 3. The flow starts with data import, and removes null value (RNV) to ignore the other mobile operator users, then removes extreme value (REV) to eliminate the spam call, and then builds graph and runs PageRank on the graph, finally list the users in order of decrease PageRank value.

The running result proved that the ETL operation in MapReduce can handle 1.3GB call information record and the extracted graph data is about 150MB, and the whole operation takes 2633seconds on our 4 computing nodes cluster. The single BSP platform can't handle the original size of data, and the single MapReduce platform can't finish computing so fast.

5 Performance

We have tested the MPGM for its functionality, reliability, usability, efficiency, maintainability, and portability. The evaluation was performed on clusters having 9 nodes, where each node consists of 2 Intel(R) Xeon(R) CPU E5530, 48 GB main memory and 1024 GB hard drive. The evaluation data is a randomly generated graph data set scale ranging from 10,000 edges to 2000,000 edges. We also deployed a BC-PDM on the same cluster and run some social network analysis algorithms using Google web data. Some of the results are presented in Figure 4. Finally, we compared MPGM and BC-BSP with the PageRank algorithm on a 4-node cluster, but where the nodes have the same hardware. The results are recorded in Figure 5. The characteristics of these graphs' data are shown in Table 1.

Table 1. Networks Basic Structural Properties

Name	|N|	|E|	Type
data_set_1	17500	100000	Random
data_set_2	72000	500000	Random
data_set_3	175000	1000000	Random
data_set_4	720000	5000000	Random
data_set_5	1750000	10000000	Random
data_set_6	3500000	20000000	Random
GoogleWeb	875713	5105039	Web

Table 2 show that most graph mining jobs can be accomplished in a short time and benefit from well-designed architecture. Also, the MPGM has a higher performance than BC-PDM and BC-BSP.

Table 2. Runtime of some Graph Mining Algorithm Components(Second)

Graph Data Set	Eigenvector Centrality Measure	InDegree Count	MSP	PageRank	Closeness Centrality	Personal Centrality	Clustering Coefficient	RWR
data_set_1	16.2	13.2	166.1	25.2	31.2	13.1	13.1	22.2
data_set_2	25.1	16.2	310.1	40.2	70.5	16.1	19.1	28.1
data_set_3	28.2	22.1	343.0	61.5	31.4	19.1	19.2	37.2
data_set_4	88.2	64.2	696.1	202.4	25.4	22.0	31.1	169.2
data_set_5	173.6	73.2	995.2	439.6	32.1	31.2	55.2	313.4
data_set_6	643.7	199.3	1278.3	1241.6	55.7	52.2	151.2	688.7
GoogleWeb	199.2	79.2	721.3	304.6	31.7	34.2	52.2	331.5

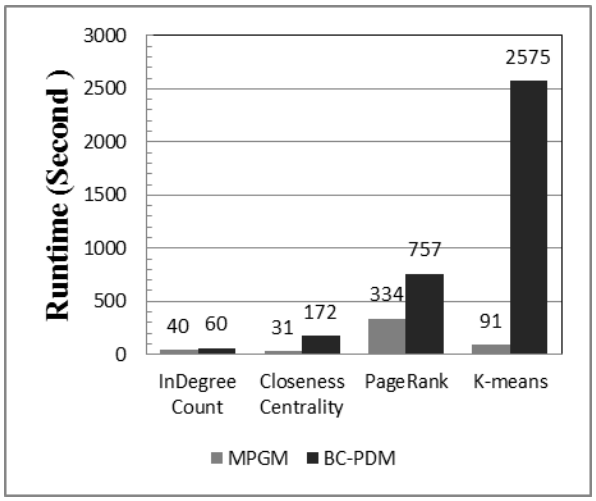

Figure 4: Comparison of MPGM and BC-PDM on data_set_5

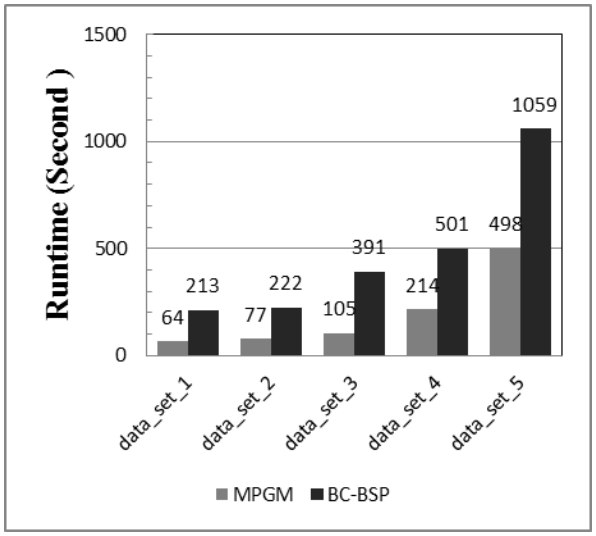

Figure 5: PageRank Performance of MPGM and BC-BSP

6 CONCLUSION

In this study, we introduced MPGM based on Cloud computing. It has the ability to analyze big graph data and achieved a better performance than the Hadoop-based data mining tools BC-PDM and BSP-based parallel platform BC-BSP. We expected to mix more parallel computing model to achieve a higher performance of graph mining both in data scale and computing speed.

7 Acknowledgment

This work is supported by the National Key Basic Research and Department (973) Program of China (No.2013CB329603) and the National Science Foundation of China (Nos.61375058,and 71231002). This work is also supported by the Special Coconstruction Project of Beijing Municipal Commission of Education.

8 References

[1] J. Dean, and G. Sanjay, MapReduce: Simplified data processing on large clusters, Communications of the ACM., vol. 51, no. 1, pp. 107-113,2008.

[2] S. Marc, S. W. Otto, D. W. Walker, J. Dongarra, and S. Huss-Lederman, MPI: The Complete Reference. MIT press, 1995.

[3] L. G. Valiant, A bridging model for parallel computation, Communications of the ACM., vol. 33, no. 8, pp. 103-111, 1990.

[4] S. Owen, A. Robin, T. Dunning, and E. Friedman, Mahout in Action. Manning, 2011.

[5] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein, Graphlab: A new framework for parallel machine learning, arXiv preprint., arXiv:1006.4990, 2010.

[6] U. Kang, C. E. Tsourakakis, and C. Faloutsos, Pegasus:A peta-scale graph mining system implementation and observations, in Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on, 2009.

[7] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, Dryad: Distributed data-parallel programs from sequential building blocks, ACM SIGOPS Operating Systems Review, vol. 41, no. 3, pp. 59-72, 2007.

[8] L. Yu, J. Zheng, W. Shen, B. Wu, B. Wang, L. Qian, and B. Zhang, BC-PDM: Data mining, social network analysis and text mining system based on cloud computing, presented at the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012.

[9] G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I.Horn, N. Leiser, and G. Czajkowski, Pregel: A system forlarge-scale graph processing, in Proceedings of the 2010ACM SIGMOD International Conference on Management of Data, 2010.

[10] Y. Bao, Z. Wang, Y. Gu, G. Yu, F. Leng, H. Zhang, B. Chen, C. Deng, and L. Guo, BC-BSP: A BSP-based parallel iterative processing system for big data on cloud architecture, in Proc.Database Systems for Advanced Applications, Springer Berlin Heidelberg, 2013.

[11] S. Seo, E. J. Yoon, J. Kim, S. Jin, J. Kim, and S. Maeng, Hama: An efficient matrix computation with the mapreduce framework, in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, 2010.

[12] L. Page, S. Brin, R. Motwani, and T. Winograd, The PageRank citation ranking: Bringing order to the web, 1999.

[13] http://malt.ml.cmu.edu/mw/index.php/Random walk with restart.

[14] J. M. Kleinberg, Authoritative sources in a hyperlinked environment, Journal of the ACM (JACM), vol. 46, no. 5, pp. 604-632, 1999.

[15] M. Girvan and M. E. J. Newman, Community structurein social and biological networks, Proceedings of theNational Academy of Sciences, vol. 99, no. 12, pp. 7821-7826, 2002.

[16] A. Clauset, M. E. J. Newman, and C. Moore, Finding community structure in very large networks, Physical Review E, vol. 70, no. 6, 2004.

[17] M. E. J. Newman and M. Girvan, Finding and evaluating community structure in networks, Physical Review E, vol. 69, no. 2, 2004.

[18] U. N. Raghavan, R. Albert, and S. Kumara, Near linear time algorithm to detect community structures in largescale networks, Physical Review E, vol. 76, no. 3, 2007.

[19] Z. Zeng, B. Wu, and H. Wang, A parallel graph partitioning algorithm to speed up the largescale distributed graph mining, in The 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, 2012.

[20] G. Karypis and V. Kumar, Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0, 1995.

[21] J. Yang and D. Zhang, Lightweight workflow engine based on Hadoop and OSGI, presented at the 5th IEEE International Conference on Broadband Network & Multimedia Technology, Beijing, China, 2013.